SC22
Dallas, TX | hpc accelerates.

Hybrid AI/HPC Approaches and Linear Algebra

Nahid Emad

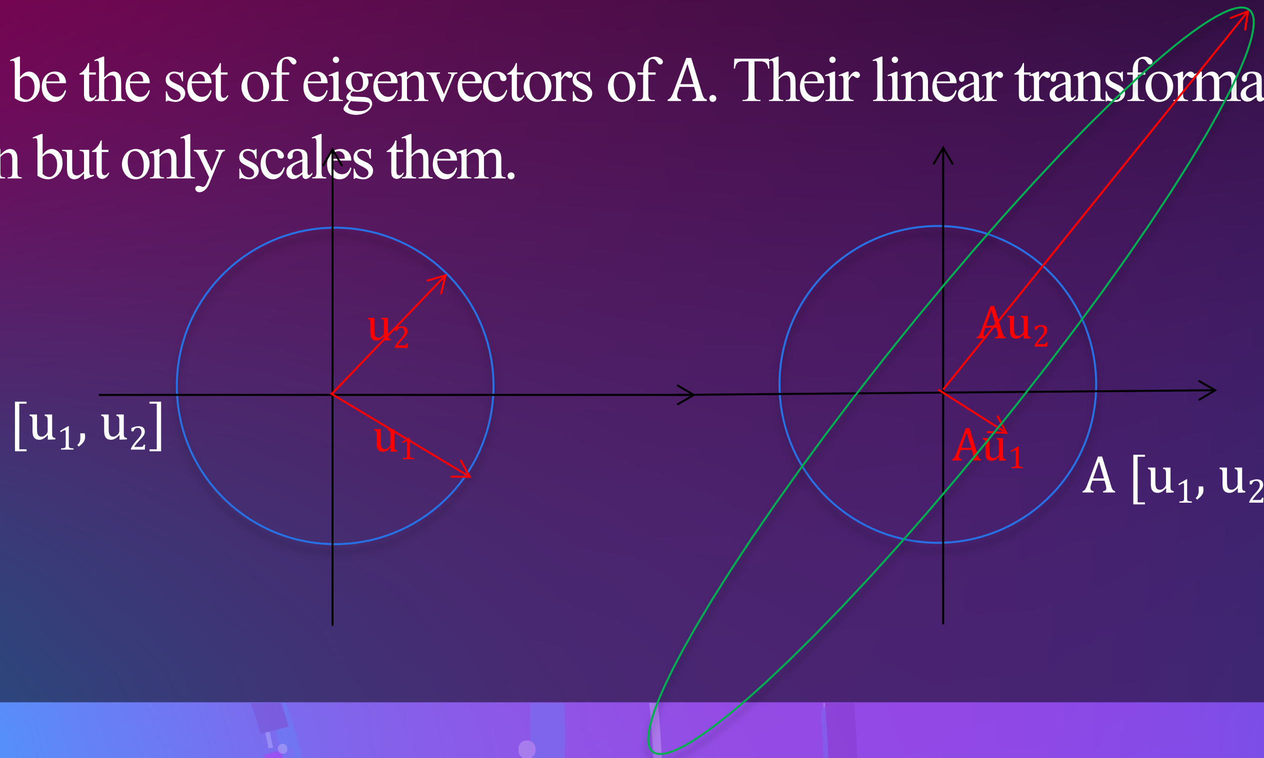
University of Paris Saclay / UVSQ
Maison de la Simulation & LI-PaRAD

Linear algebra main problem in ML/DL

- In machine learning, many problems can be solved by **linear transformations** and systems of linear equations.
- Let A and Y be n-size matrix representing a set of n observations and the vector of their labels. The search of a function f(A)=Y can be expressed as a linear system:

$$Ax=Y$$

- Let $(u_1, \ldots, u_n)$ be the set of eigenvectors of A. Their linear transformation by A does not change their orientation but only scales them.



$$A\,[u_1, u_2] = \begin{pmatrix} 0.5 & 0 \\ 0 & 3 \end{pmatrix} [u_1, u_2]$$

Dominant eigenspace in ML

Principal Component Analysis: The goal is to find an orthonormal basis of the space of a dataset such that the variance of the dataset (degree of dispersion) in this basis is maximized. PCA helps reduce redundancies in datasets and extract important features while preserving accuracy.

- Let $X \in \mathbb{R}^{n\times p}$ be a centered matrix of n observations of p features. The PC of X are the dominant eigenvectors of its covariance matrix $A = \frac{1}{n}X^TX$.
- The PC of X are its dominant right singular vectors: $X = U\Sigma V^T$ with $U \in \mathbb{R}^{n\times n}, V \in \mathbb{R}^{p\times p}$ unitary and $\Sigma \in \mathbb{R}^{n\times p}$ diagonal matrices of singular values. $A = X^TX = V\Sigma^2V^T$. The columns of V are the right singular vectors of X.

PageRank algorithm example: The Markov matrix leads to the equation which the steady state depends on one dominant component: $\lambda_1^k \boldsymbol{u_1} + \alpha_1 \cancel{\lambda_2^k u_2 + \ldots + \alpha_n \lambda_1^k} u_n$.

ML methods and linear algebra

Goal: Build smarter machines thinking and acting on their own (needs of training –still- and more and more data)

- Supervised machine learning methods
  - Linear regression, logistic regression, **recommendation systems**, ANN, etc.
  - Linear algebra problem as linear systems and **eigenproblems**
- Unsupervised machine learning methods
  - K-means for partitioning, **dimensionality reduction**, **CPA**, etc.
  - Essentially **eigenproblems** and **SVD**
- Reinforcement learning methods (exploration & exploitation)
  - Bandit, **Markovian decision problems**, game trees.

High performance data analysis

- **Data production** is now faster than compute capabilities
- **Applications** are classical simulation, social network-based simulation, ML algorithms
- Emerging **Exascale supercomputers** : Multi-level architectures (processor, memory, …), mixed arithmetic (16, 32, 64 bits,...), ..., and convergence of distributed and parallel computing inside them.
- Need of new **programming paradigms** for this extreme computational and data sciences programming.
- **New methods** must be developed (involving applied math, graph theory, Bayesian network, statistic, linear algebra, game theory, ...) but also, the new approaches such as transformer used in NLP.
- **Big Data analysis and HPC convergence** is crucial to propose future machine learning algorithm for Post-Petascale platforms and supercomputers

New paradigms for new intelligent applications

Outline

- Main problems in linear algebra (moderate size)
- Large and sparse linear algebra problem
- High-performance AI and LA with applications
- Concluding remarks

Outline

- Main problems in linear algebra (moderate size)
- Large and sparse linear algebra problem
- High-performance AI and LA with applications
- Concluding remarks

Main problems in linear algebra (moderate size)

Linear system (LS) :

$\boxed{\text{Let } \mathrm{A} \in \mathbb{C}^{n\times n},\ \mathrm{b} \in \mathbb{C}^{n},\ \text{find } \mathrm{x} \in \mathbb{C}^{n},\ \text{such that} : A.x = b}$

Eigenproblem (EIG) :

$\boxed{\text{Let } \mathrm{A} \in \mathbb{C}^{n\times n},\ \text{find } \lambda_i \in \mathbb{C} \text{ and } u_i \in \mathbb{C}^{n} \text{ such that} : A.u_i = \lambda_i.u_i \quad (i = 1, \dots, n)}$

- Solving LS (topic well mastered overall)
  - **Direct** methods as Gauss and Gauss-Jordan, Cholesky, Householder based on LU, Cholesky, QR decomposition.
  - **Iterative** methods as Jacobi, Gauss-Seidel, Relaxation.
- Solving EIG (**topic not so well mastered**)
  - **Only iterative** methods (Abel-Ruffini theorem) as Jacobi and QR

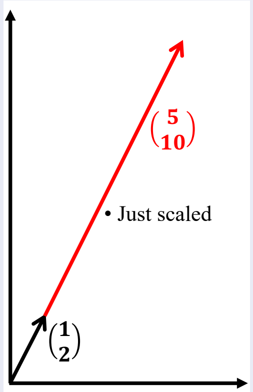
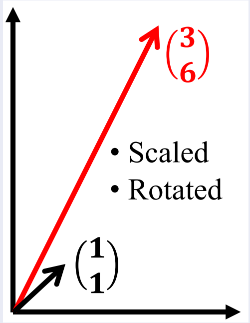
Focus on Eigenproblem

Eigenproblem (EIG) :

Let A $\in \mathbb{C}^{n\times n}$, find $\lambda_i \in \mathbb{C}$ and $u_i \in \mathbb{C}^n$ such that : $A.u_i = \lambda_i.u_i \quad (i = 1, \dots, n)$

Power of eigenvectors :

- ✓A doesn't change the orientation of an eigenvector and/or eigenspace but just scales it.
- ✓Principal components or axes of dataset.

A	x	Ax
$\begin{pmatrix}1 & 2\\ 2 & 4\end{pmatrix}$	$\begin{pmatrix}1\\2\end{pmatrix}$	
	$\begin{pmatrix}1\\1\end{pmatrix}$	

Eigen-elements of A : λ_1=0, λ_2=5 and $v_1=\begin{pmatrix}-2\\1\end{pmatrix}$, $v_2=\begin{pmatrix}1\\2\end{pmatrix}$

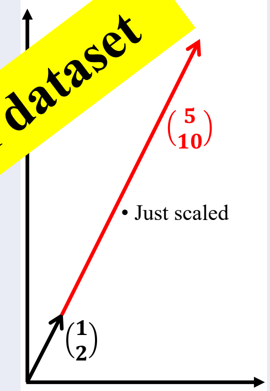
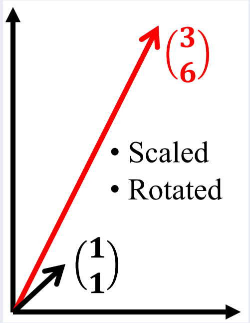
Focus on Eigenproblem

Eigenproblem (EIG) :

$$\boxed{\text{Let } A \in \mathbb{C}^{n\times n}, \text{ find } \lambda_i \in \mathbb{C} \text{ and } u_i \in \mathbb{C}^n \text{ such that} : A.u_i = \lambda_i.u_i \quad (i = 1, \dots, n)}$$

Power of eigenvectors :

- ✓ A doesn't change the direction of an eigenvector and/or eigenspace but just scales it.
- ✓ Principal components or axes of dataset.

A	x	Ax
$\begin{pmatrix} \mathbf{1} & \mathbf{2} \\ \mathbf{2} & \mathbf{4} \end{pmatrix}$	$\begin{pmatrix} 1 \\ 2 \end{pmatrix}$	
	$\begin{pmatrix} 1 \\ 1 \end{pmatrix}$	

The backbones of dataset

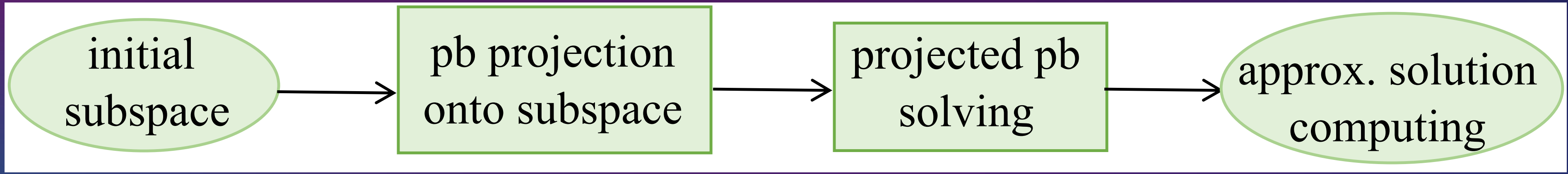
Eigen-elements of A : λ_1=0, λ_2=5 and $v_1=\begin{pmatrix} -2 \\ 1 \end{pmatrix}$, $v_2=\begin{pmatrix} 1 \\ 2 \end{pmatrix}$

Outline

- Main problems in linear algebra (moderate size)
- Large and sparse linear algebra problem
- High-performance AI and LA with applications
- Concluding remarks

Large and sparse linear algebra problems

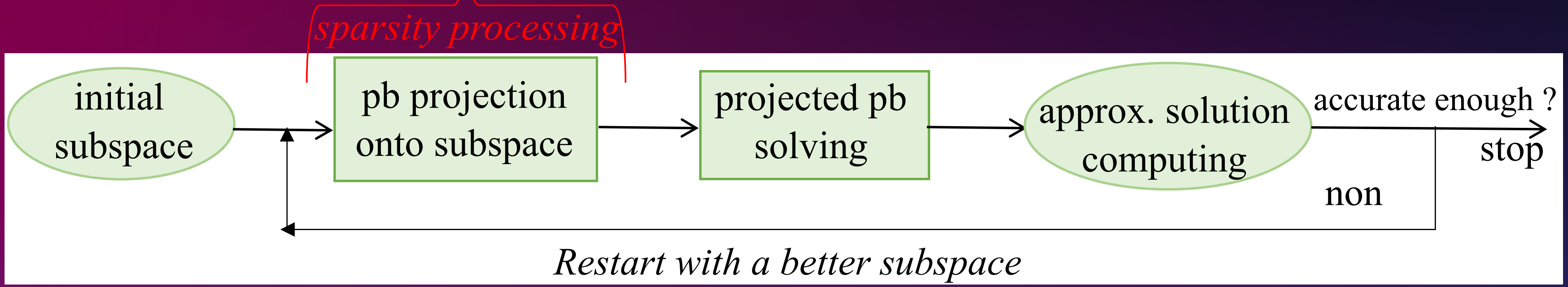
- **Sparse dataset**
  - ➢ Avoiding fill-in – iterative methods
  - ➢ Problem : how to compress the dataset ? Use of ML methods
- **Large dataset**
  - ➢ Dimensionality reduction - projection onto Krylov subspace
  - ➢ Problem : how to choose the projection subspace ? Too large/small-size, …



Large and sparse linear algebra problems

Iterative projection method

- Preserve sparsity
- Reduce the problem size



Main problems for these methods

- Sparsity processing
- Krylov subspace: better choice of v for $\mathbb{K}_m(A, v) = span\ (v, Av, \ldots, A^{m-1}v)$

What about m ?

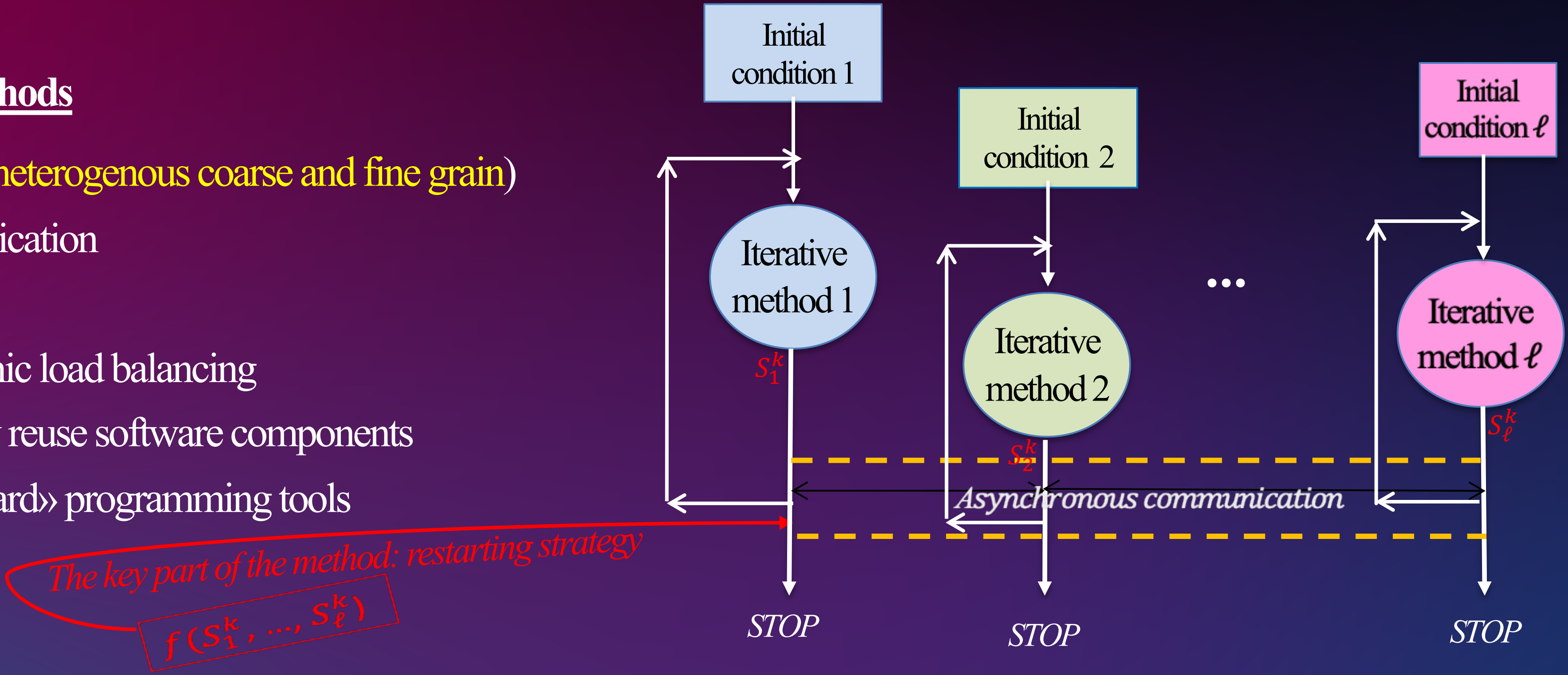
better choice of m and v ?

Unite and Conquer methods - an innovative approach

Suppose we have ℓ iterative methods to solve the same given problem. The unite and conquer approach consists of making collaborate these ℓ methods in order to accelerate the convergence of the whole system.

Characteristics of UC methods

- Multi level parallelism (heterogenous coarse and fine grain)
- Asynchronous communication
- Fault tolerance
- Great potential to dynamic load balancing
- Many parameters, many reuse software components
- Need well suited «standard» programming tools



Well suited to large-scale computing systems

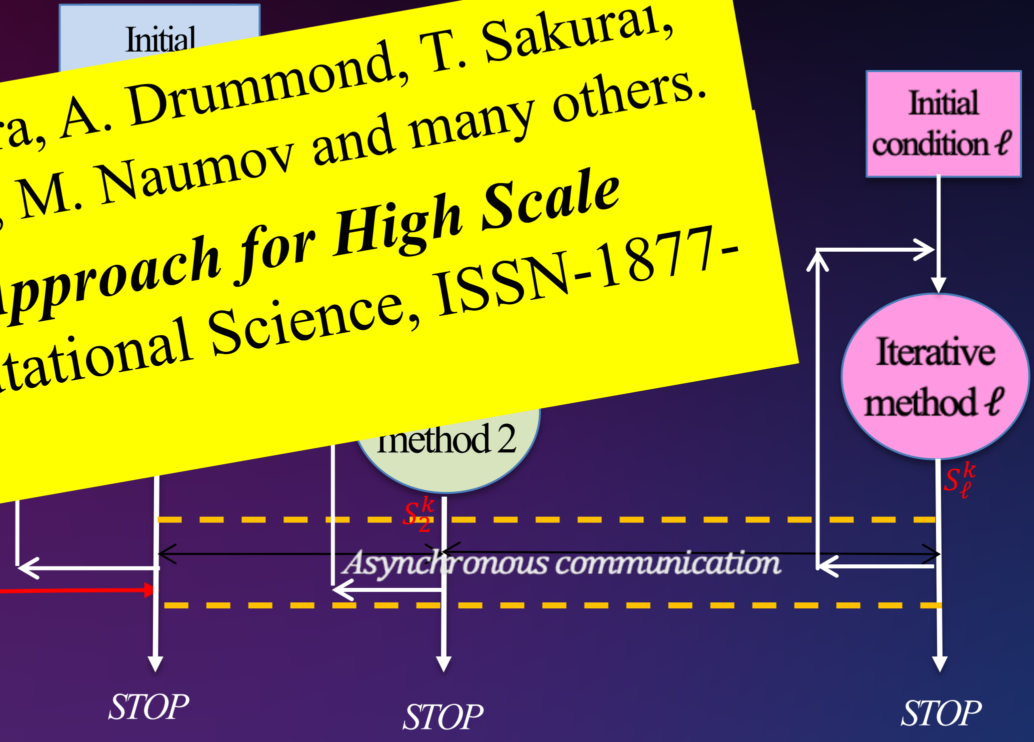
Unite and Conquer methods - an innovative Approach

Suppose we have ℓ iterative methods to solve the same given problem. The unite and conquer approach consists of making collaborate these ℓ methods in order to accelerate the convergence of the whole system.

Characteristics of UC methods

- Multi level parallelism (heteroge…
- Asynch…
- F…
- G…
- Ma…
- Nee… …g tools

With S. Petiton, G. Edjlali, A. Fazeli, J. Dongarra, A. Drummond, T. Sakurai, M. Tsuji, C. Calvin, A. Fender, Z. Liu, J. Eaton, M. Naumov and many others.
N. Emad, S. Petiton. ***Unite and Conquer Approach for High Scale Numerical Computing***, Journal of Computational Science, ISSN-1877-7503, 2016.



The key part of the method: restarting strategy
$f(S_1^k, \dots, S_\ell^k)$

Well suited to large-scale computing systems

Unite and Conquer methods

Due to the numerical and computational properties of a UC method, its overall convergence and computational performance are better than that of each of its co-methods individually.

- Multiple-Method : Case of UCM when the co-methods are the instances of the same iterative method. Example: MERAM, MIRAM, MIRLanczos, with different or nested subspaces.
- The asynchronism of communications implies better computational performance but introduces a certain *non-determinism*.
- The application of the UC approach to ML methods, which are inherently non-deterministic, does not suffer from this non-determinism.

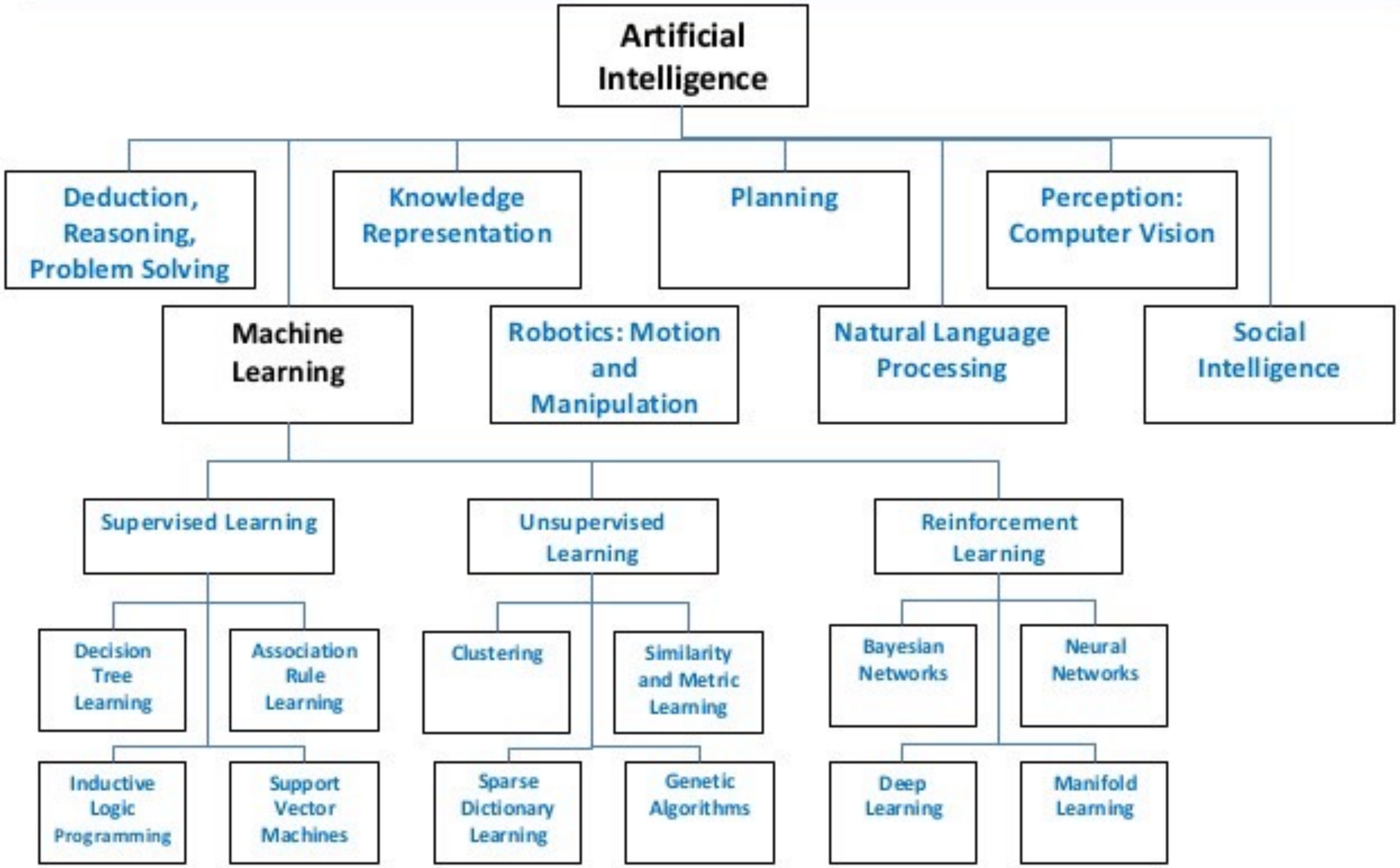
Outline

- Main problems in linear algebra (moderate size)
- Large and sparse linear algebra problem
- High-performance AI and LA with applications
- Concluding remarks

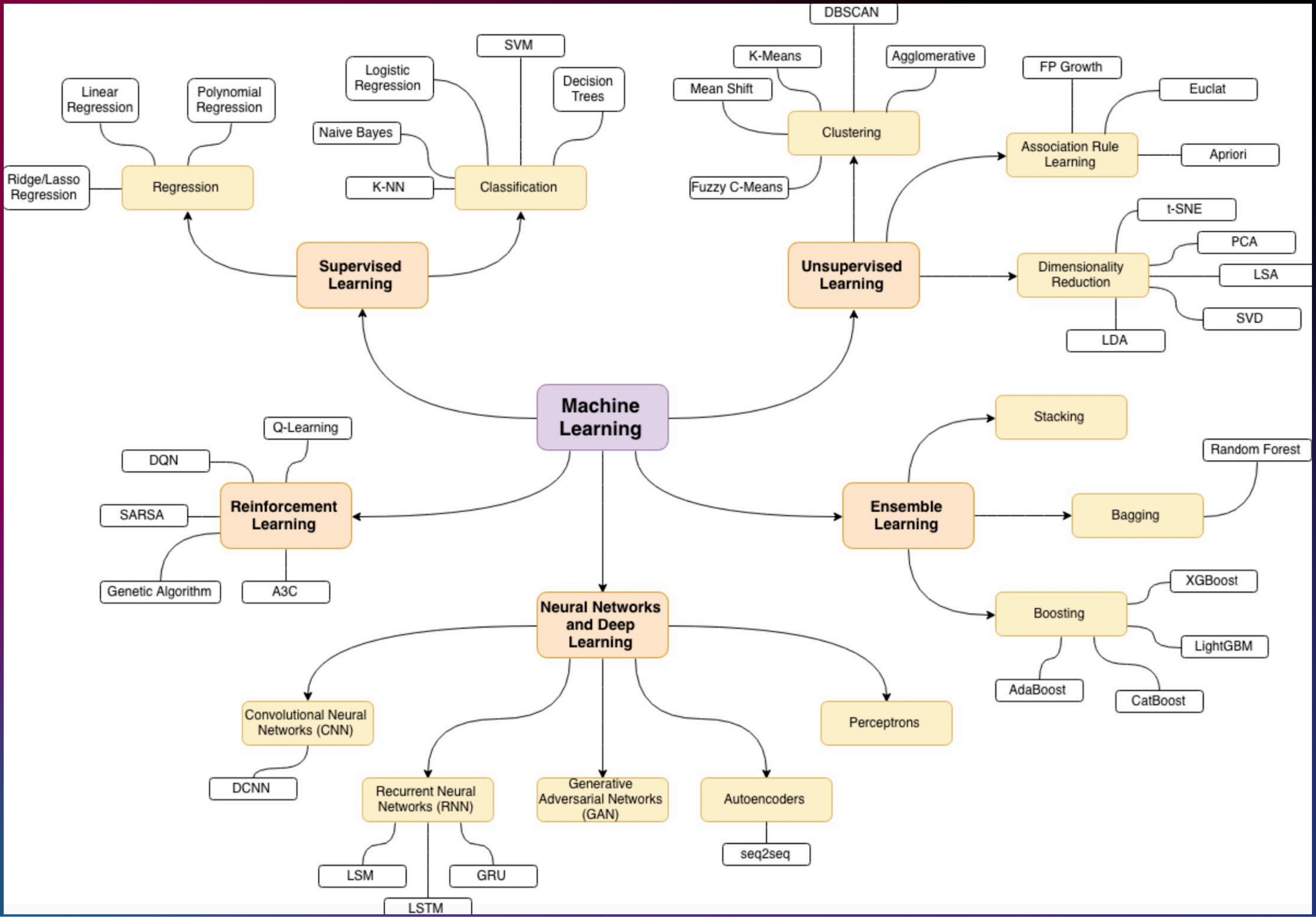
HPC and AI convergence

- About ML/DL:
- **1943** : first NN
- **1957** : first NN with training
- **1974-1981** : “silence”
- **1981** : first perceptron multilayer
- **1990** : first CNN-LeCun
- **1997** : first RNN (LSTMs)
- **Circa 2012** : The flight of ML/DL with **Big Data and computing power**
- **Circa 2017 :** Extension of NLP with transformers

Artificial Intelligence / Machine Learning Classification



http://image.slidesharecdn.com/deepdiveinaimlventurelandscape-150831132221-lva1-app6891/95/deepdive-in-aiml-venture-landscape-by-ajit-nazre-rahul-garg-3-638.jpg?cb=1441027412



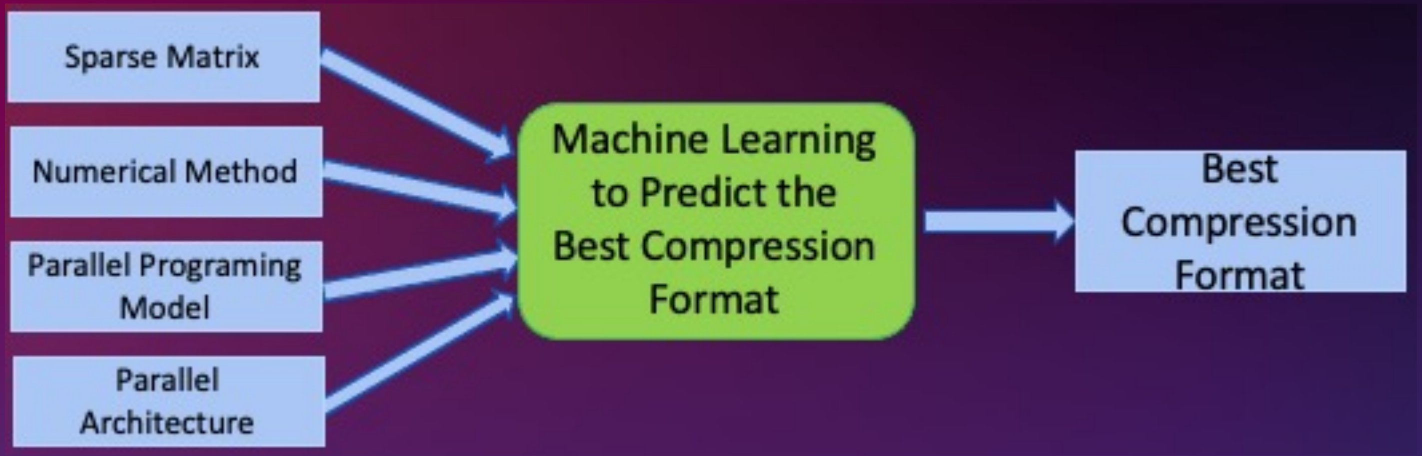
https://raw.githubusercontent.com/trekhleb/homemade-machine-learning/master/images/machine-learning-map.png

High performance AI and LA with applications

1. Sparse computation : a common topic of HP LA & ML/DL (supervised ML)
2. Focus on clustering (unsupervised ML) using UC methods
   - K-means and spectral computation
   - nvGraph of NVIDIA (https://github.com/rapidsai/nvgraph/blob/main/cpp/src/lanczos.cu)
3. Focus on (semi)supervised (classification) using UC methods
   - UCM application to ensemble learning
   - UCEL framework (the version for behavior profiling integrated to an alarm system in Atos company)

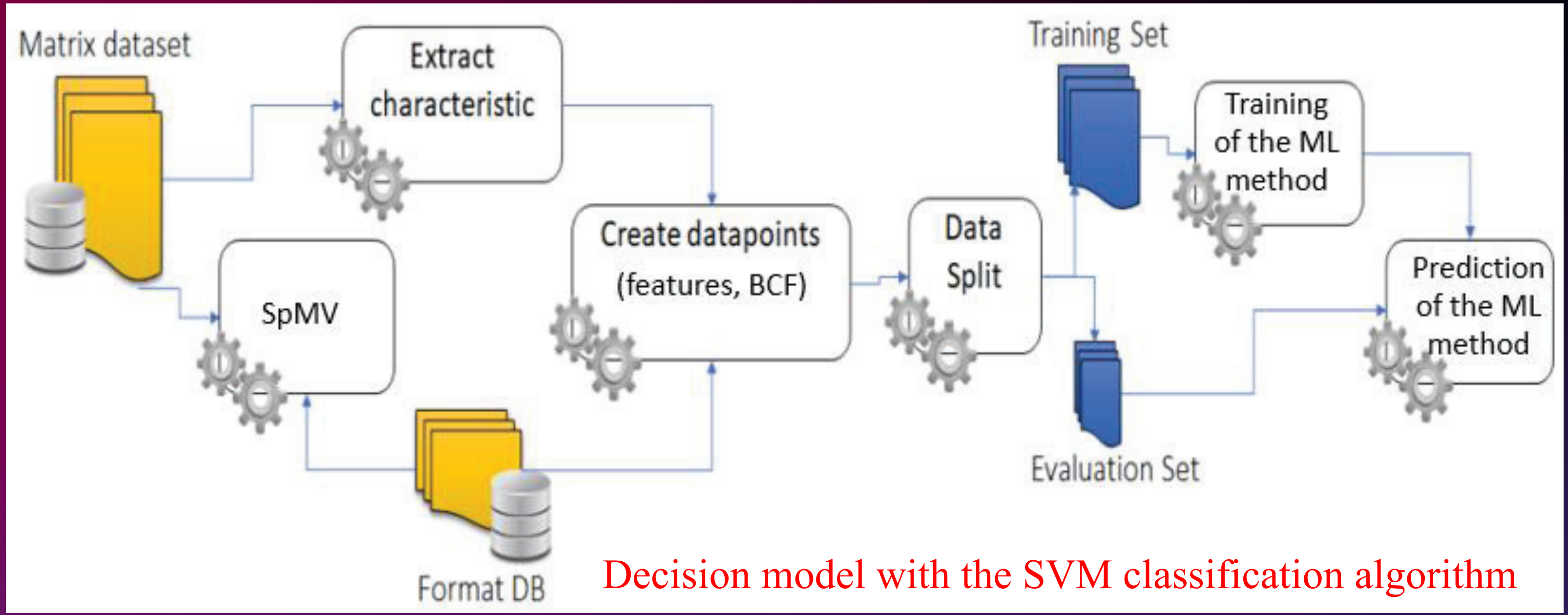
Sparse Computation : A common topic of HP LA & ML/DL

Automatic detection of the best sparse compression format as a function of the context (numerical method, parallel programming model, parallel/distributed architecture, etc.) : Auto-Tuning, Expert System and then Machine Learning



Features: number of rows, nonzero elements, matrix density, the matrix is unstructured, is structured (diagonal, triangular, band, etc.), maximum/minimum number of nonzero elements per row (& par column), cost of a data parallel operation, number of physical/virtual processors, …

Sparse Computation : A common topic of HPLA & ML/DL

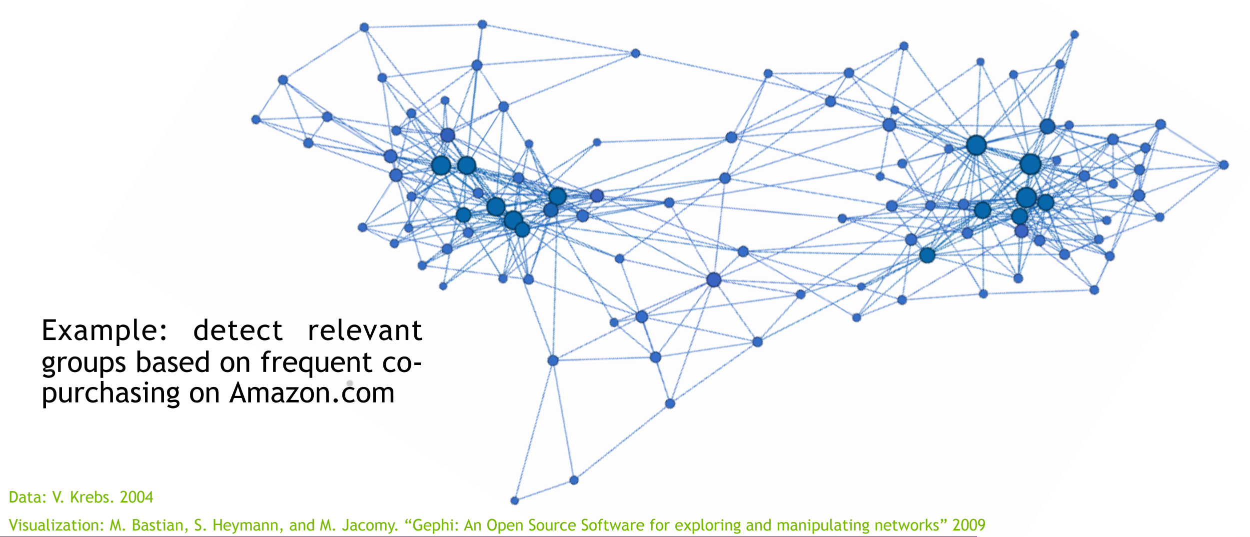


Classifier accuracy (SVM): 95.65%. **Formats:** CSR, CSC, ELL, COOC, COOR.
Hardware: Grid5K French national platform. **Labeled data:** 600

- *Mehrez,, Hamdi, Dufaud, Emad. Machine Learning for Optimal Compression Format Prediction on Multiprocessor Platform. HPCS 2018: 213-220.*
- *Hamdi et al. Machine Learning to Design an Auto-tuning System for the Best Compressed Format Detection for Parallel Sparse Computations, Parallel Process. Lett. 31(4): 2150019:1-2150019:37 (2021).*

Pipeline GPTune (https://gptune.lbl.gov/about) specifically designed for HPC applications (J. Demmel - UCB, Sherry Li, Hengrui Luo,… - LBNL).

Focus on clustering (unsupervised ML)

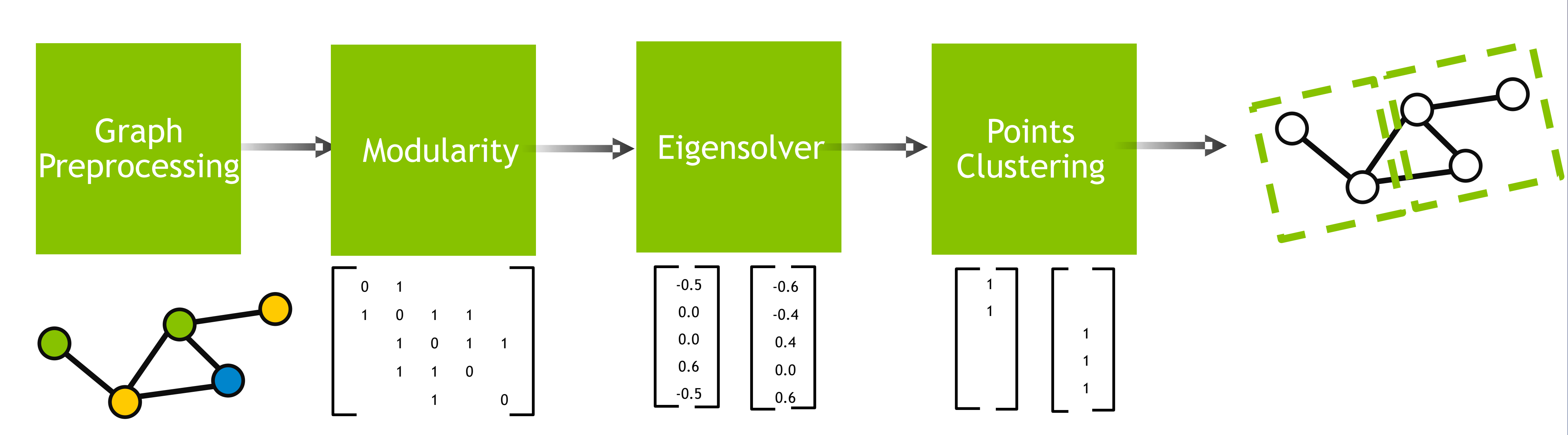


Two main methods allow partitioning vertices V of a graph $G = (V, E)$ in a set of clusters $S_k \subseteq V$ such that $V = \cup_{k=1}^{p} S_k$ are **modularity maximization** and **minimum balanced cut**. This by computing the largest eigenpairs of the modularity matrix or the smallest of the Laplacian matrix.

The multiple implicitly restarted Arnoldi MIRAMns and multiple implicitly restarted Lanczos methods MIRLanczos_ns with nested subspaces are used.

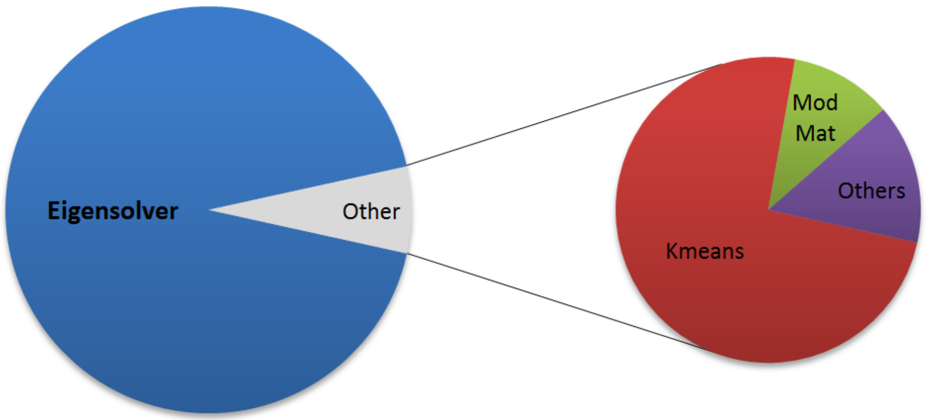
Focus on Clustering (2)

1. Let $G = (V, E)$ be an input graph and A be its weighted adjacency matrix.
2. Let p be the number of desired clusters.
3. Set the modularity matrix $B = A - \frac{1}{2\,\omega} v\, v^T$
4. Find p largest eigenpairs $BU = U\Sigma$, where $\Sigma = diag\,(\lambda_1, \dots, \lambda_p)$
5. Scale eigenvectors U by row or by column (optional).
6. Run clustering algorithm, such as k-means, on points defined by rows of U.



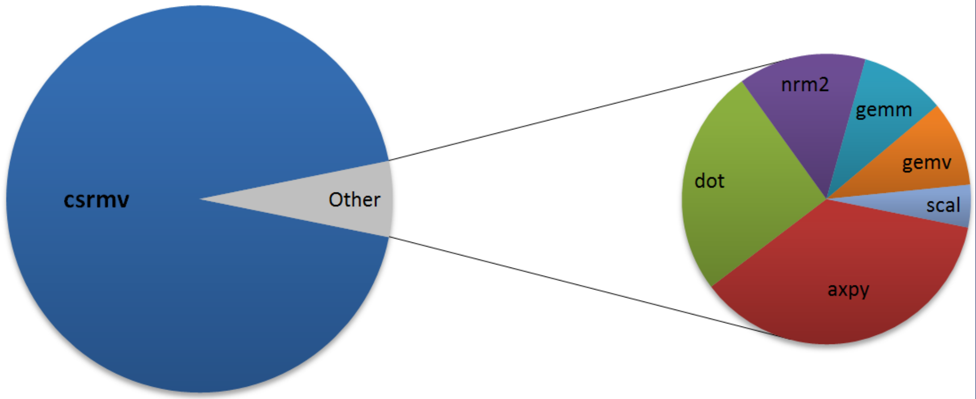
Focus on Clustering (3)

Profiling: modularity clustering



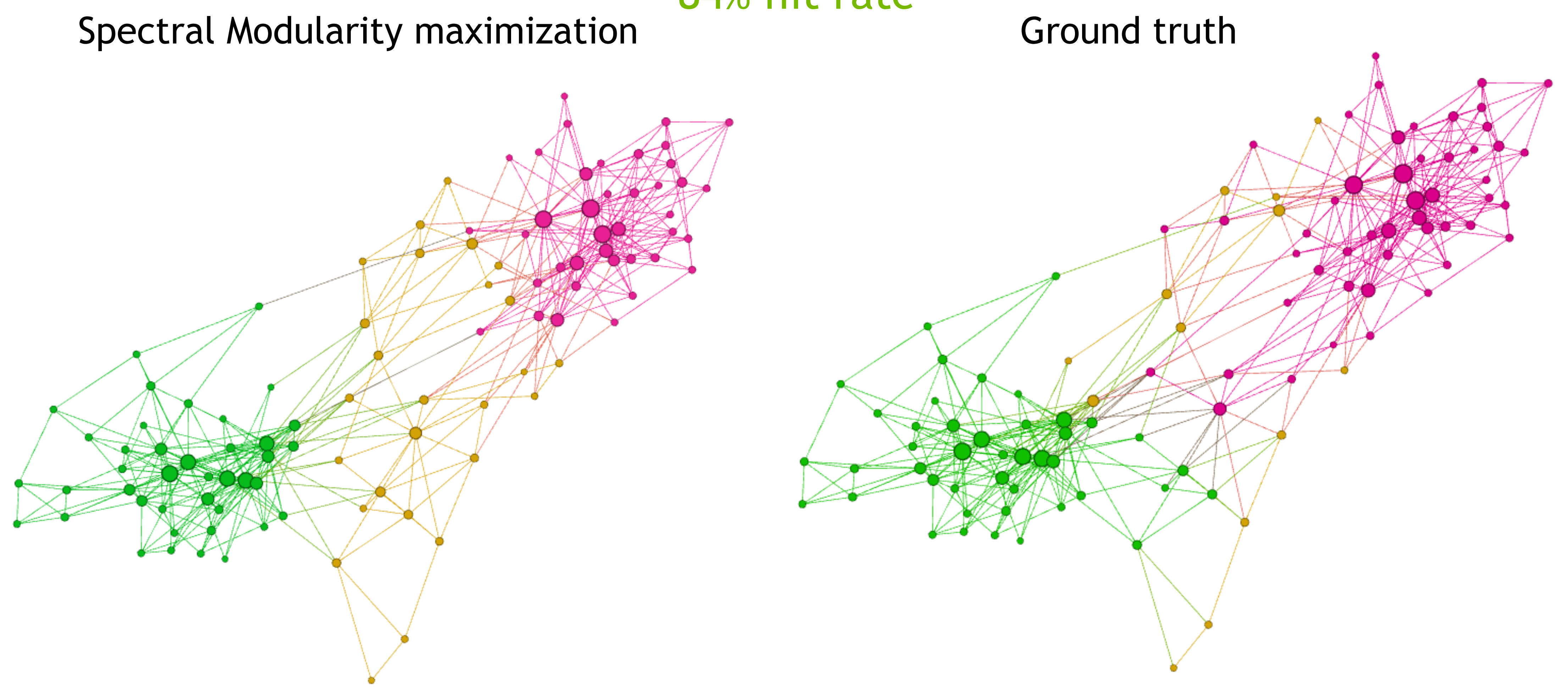
The eigensolver takes 90% of the time

The sparse matrix vector multiplication takes 90% of the time in the eigensolver



Focus on clustering (4)

84% hit rate

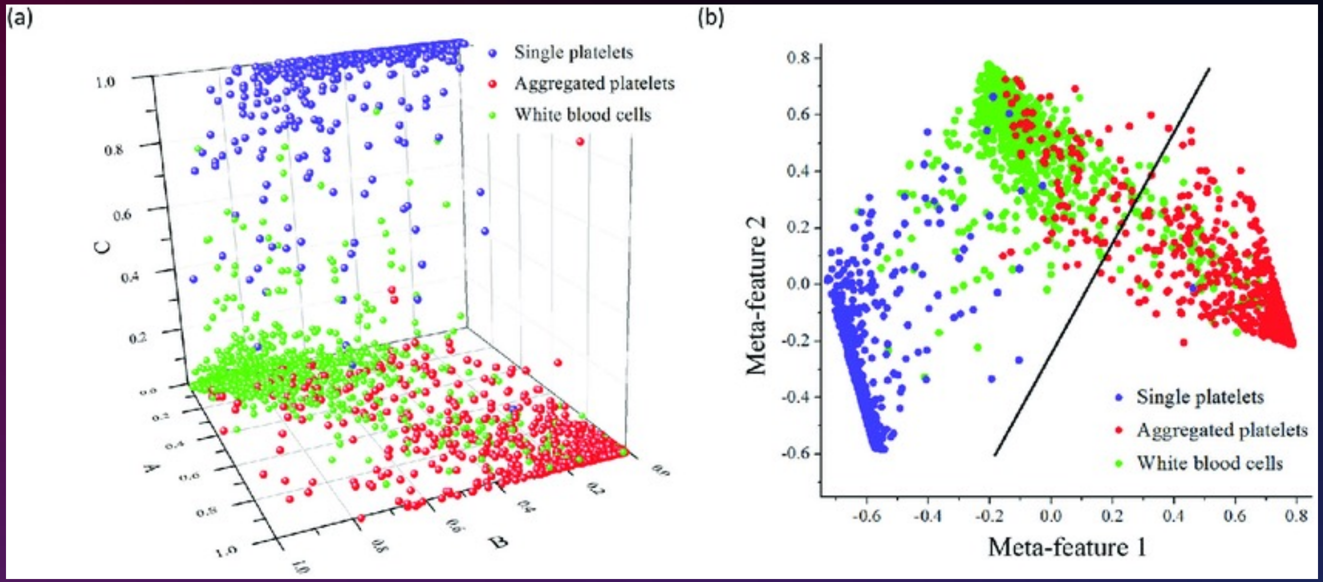


A. Fender, N. Emad, S. Petiton, M. Naumov, *Parallel Modularity Clustering*, Procedia Computer Science, Volume 108, 2017, Pages 1793-1802

Focus on (semi)supervised classification using UC methods

A process of categorizing a given dataset (structured or unstructured) into **predefined classes** (label or categories).

- Classification predictive modeling
- Binary classification
- Multi-class classification
- Multi-label classification
- Imbalanced classification



Jiang, Yiyue et al., *Label-free detection of aggregated platelets in blood by machine-learning-aided optofluidic time-stretch microscop,*Lab Chip Journal, Vol. 17, 2426-2434, The Royal Society of Chemistry, 2017.

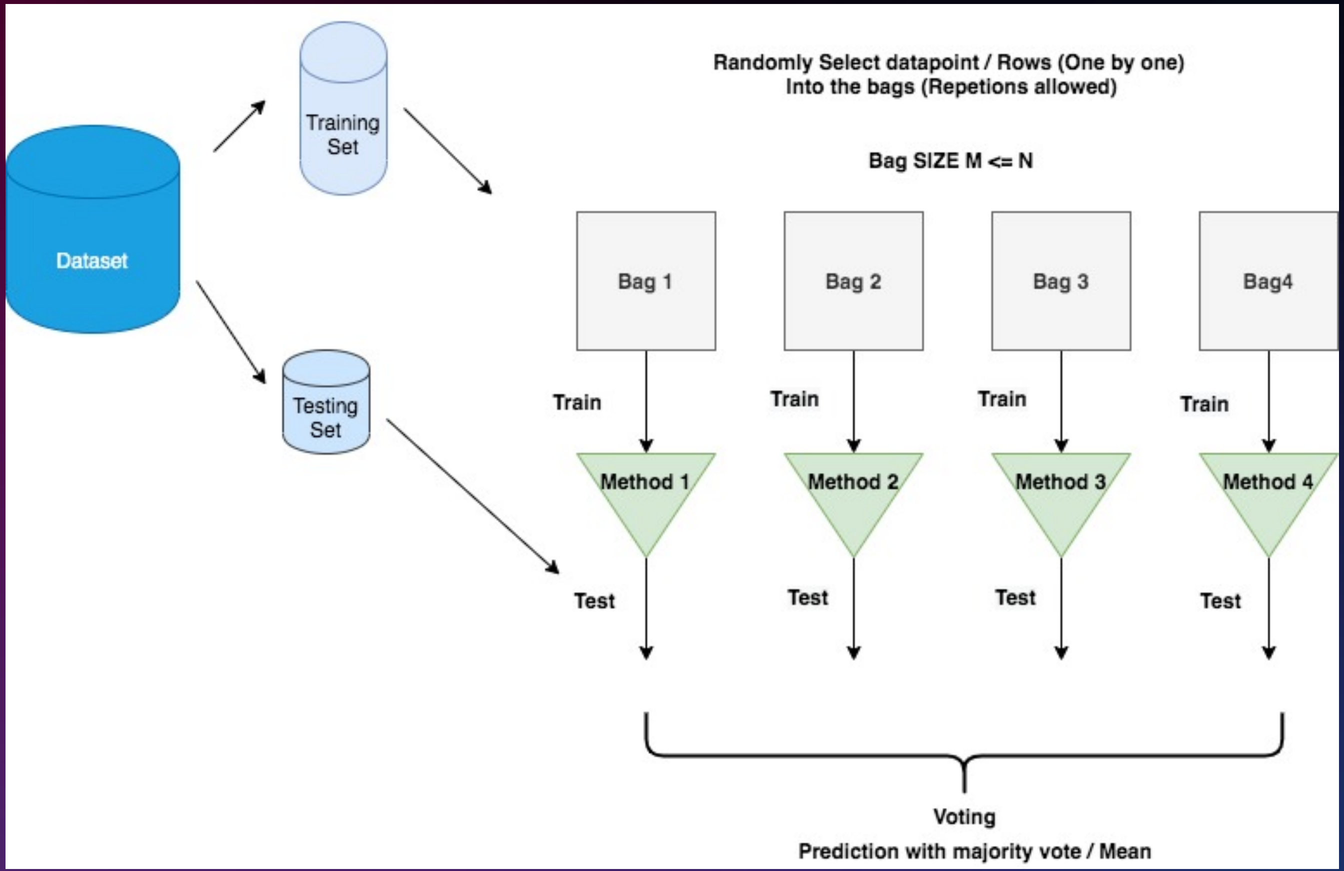
Classification methods: Logistic Regression, naive Bayes, stochastic Gradient Descent, KNN, Decision Tree, Random Forest, ANN, SVM, …

Application UC approach to ensemble learning for classification

Ensemble Learning methods

Bagging technique

1. **Start.** Choose ℓ the number of the bags and m ($\leq$n) the size of the bags.
2. **Iterate.** For $i = 1, \ldots, \ell$ do in parallel
   a)Sampling. Select the bag B_i by a random sampling technique with replacement on LD,
   b)Training and testing. Train a model L_i on the bag B_i and test $L_i$with TD dataset.
3. **Share.** On all the results of ℓ processes, use a selection system (like voting) to get the final prediction.

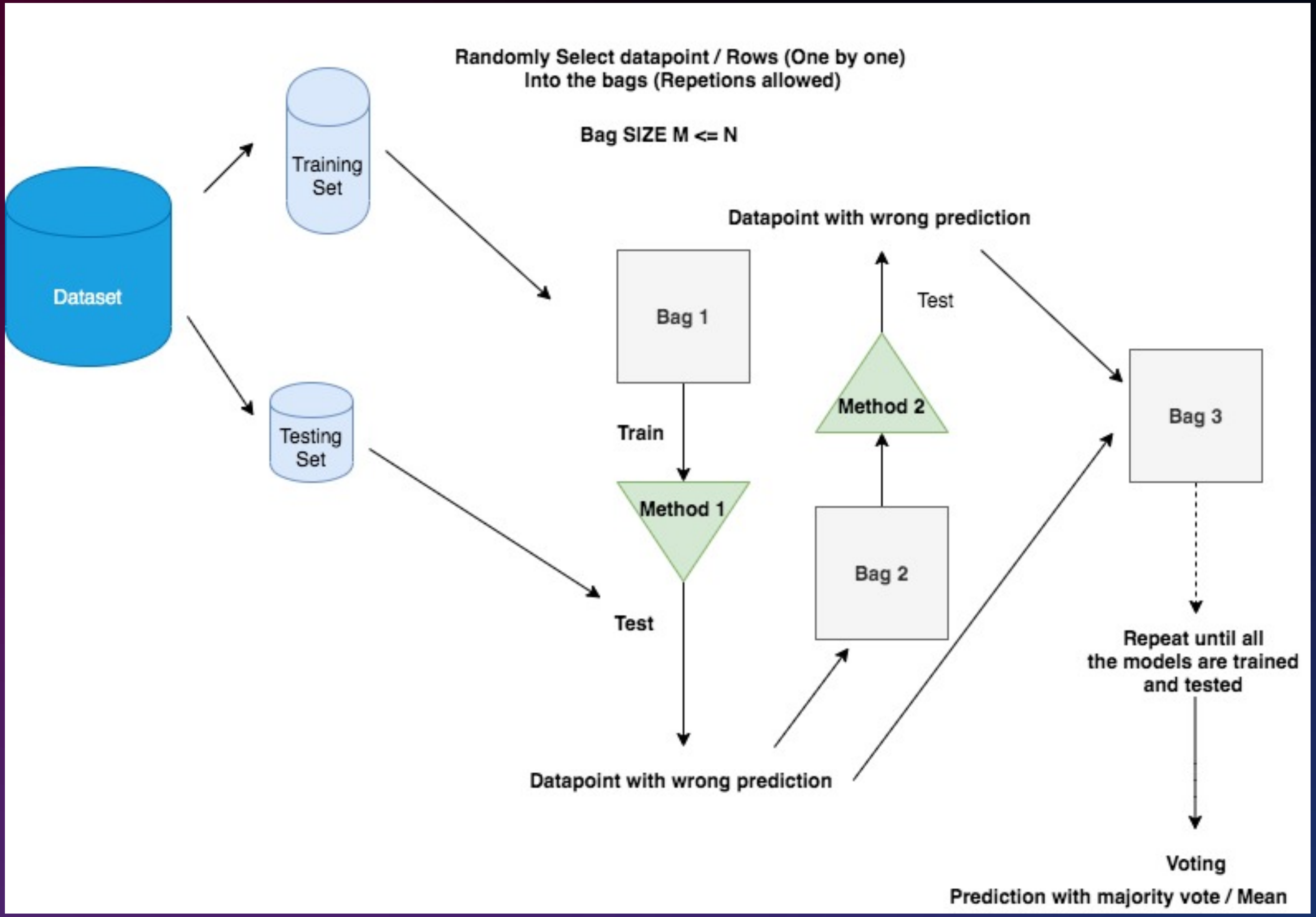


- Weak learner(s)
- Selection of the result by a voting system
- Intrinsic data parallelism

Ensemble Learning methods

Boosting technique

1. **Start.** Choose ℓ, m the number and the size of the bags, the base week leaner L_1 and define the bag B_1 =LD.
2. **Iterate.** For $i = 1, \ldots, \ell$ do
   a) *Training and testing.* Train L_i leaner on dataset B_i, produce L_i model, test it on B_i and select W_i the k_i-size miss-predicted sub-dataset of LD.
   If (P $(L_i) \geq \theta$) then put $best = i$ and stop.
   b) *Sampling.* Set the bag B_{i+1}= (1- α_i) $R_i \cup \alpha_i W_i$, where α_i is the weight given to miss-predicted data and R_i is the set of $(m - k_i)$ correctly predicted data in B_i and go to 2.
3. **Result.** Set L_{best} as a weighted combination of the previous ℓ leaners.

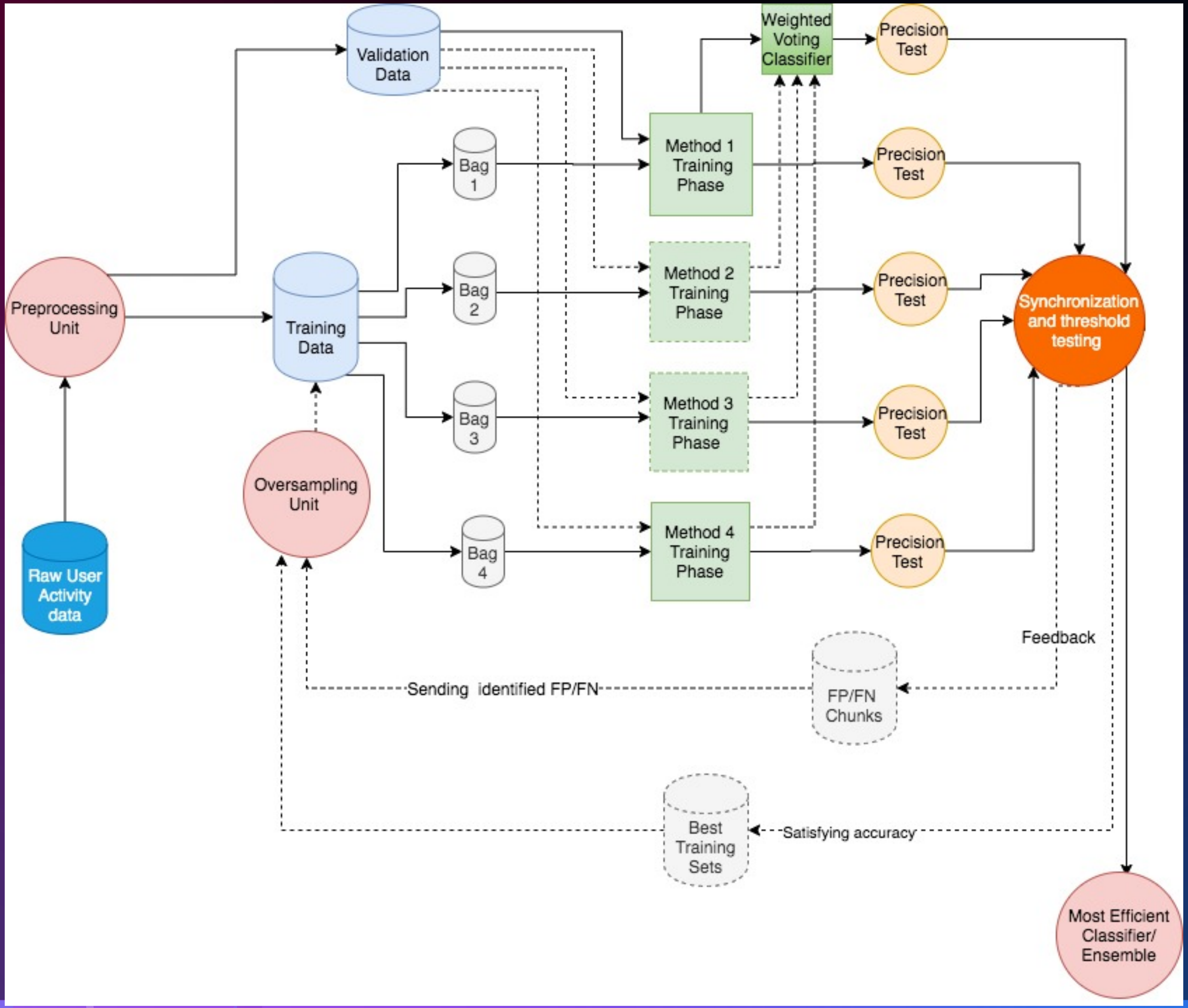


- Iterative process
- The miss-predicted data weighted more
- Selection of a weighted combination of the learners

UCEL: Unite and Conquer and Ensemble Learning

- Co-methods: Ensemble base-learners
- Partial initial parameters: Bags
- Restarting strategy is based on intermediate global result of the learners

Bagging with ℓ processes (ℓ bags & ℓ leaners) where each process is itself a boosting process with q iterations. The bagging processes cooperate in the end of each boosting iteration by exchanging information.



- *A. Diop, N. Emad, and T. Winter. A Parallel and Scalable Framework for Insider Threat Detection. In 27th IEEE International Conference HiPC, 16-19 Dec. 2020, Pune, India.*
- *A. Diop, N. Emad, and T. Winter. A Unite and Conquer Based Ensemble Learning Method for User Behavior Modeling. In 39th IEEE IPCCC Conference, Nov. 6th – 8th, 2020, Austin, Texas, USA.*

Parallel behavior profiler (in: $TD, VD, \ell, q, \theta, B_1$; out: B_{best}, L_{best})

1: **Start.** Choose ℓ, m, B^1, L^0 the ℓ bags and learners, ...

2: **Iterate.** For $i = 1, \cdots, \ell$ do *in parallel*

3: **Iterate.** For $j = 1, \cdots, q$

i^{th} computing node

4: ***Training and testing on MCN_i***
Train L_i^{j-1} on B_i^j, produce L_i^j, test L_i^j on VD and select W_i^j.

5: ***Communication: send from MCN_i to CN*** *control node*
Send $(B_i^j, L_i^j, W_i^j, \text{AUC-score}(L_i^j))$ from MCN_i to CN.

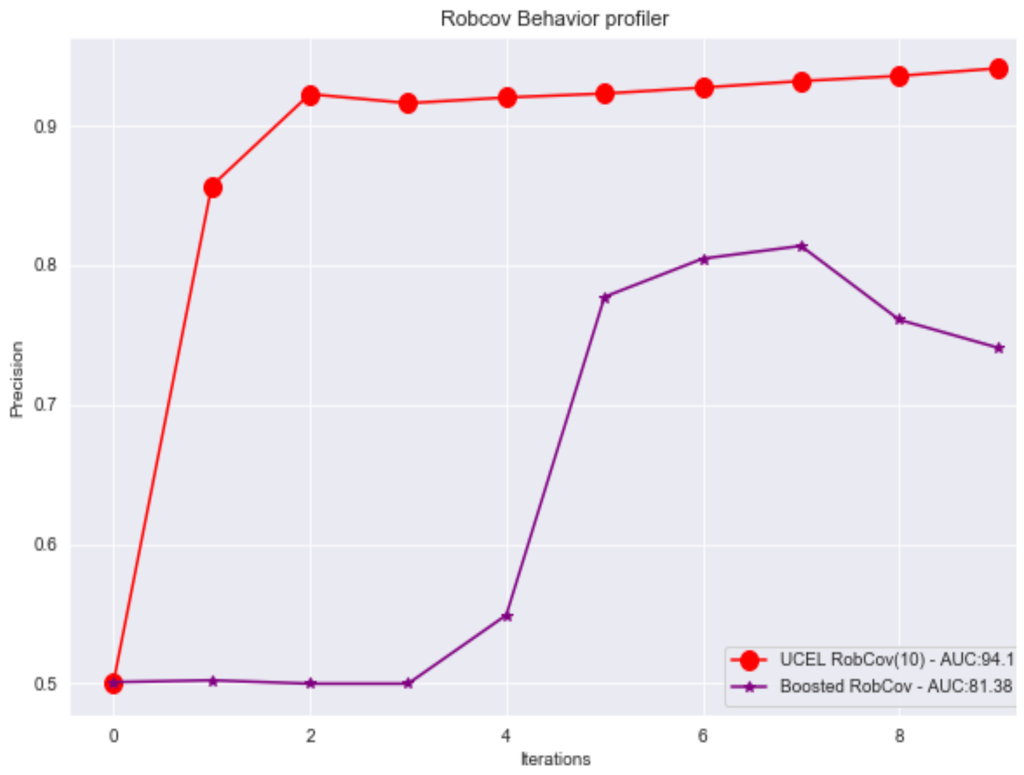
6: ***Computation and stopping test on CN***
$WVC_j = V(L_i^j, AUC(L_i^j))$
$B_{best}^j, L_{best}^j, W_{best}^j = f(L_i^j, B_i^j, W_i^j, WVC_j)$
If $(\text{AUC-score}(L_{best}^j) > \theta)$ then STOP all processes.

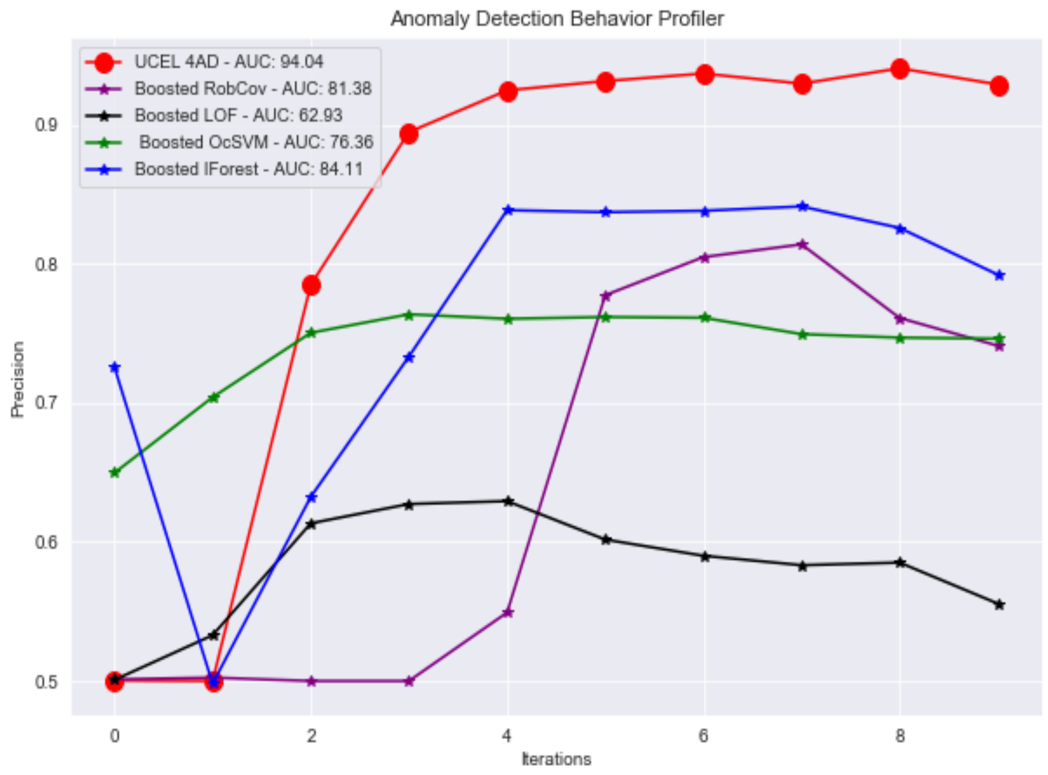
7: ***Communication: send from CN to MCN_i***
Send $(B_{best}^j, L_{best}^j, W_{best}^j)$ to all node i for $i \in [1, \ell]$.

8: ***Sampling on MCN_i***
Set the bag $B_i^{j+1} = (1-\alpha) * W_{best}^j \cup (\alpha) * R_i^j$ where R_i^j is the set of $(m_i - k_i^j)$ correctly predicted data in B_{best}^j with $k_i^j = \mathbf{card}(W_{best}^j)$ and, α is the updated weight given to miss-predicted data.

9: ***Result.***
Set L_{best} the best individual co-method or best weighted combination of co-methods during the iterations of all ℓ processes.

UCEL performance



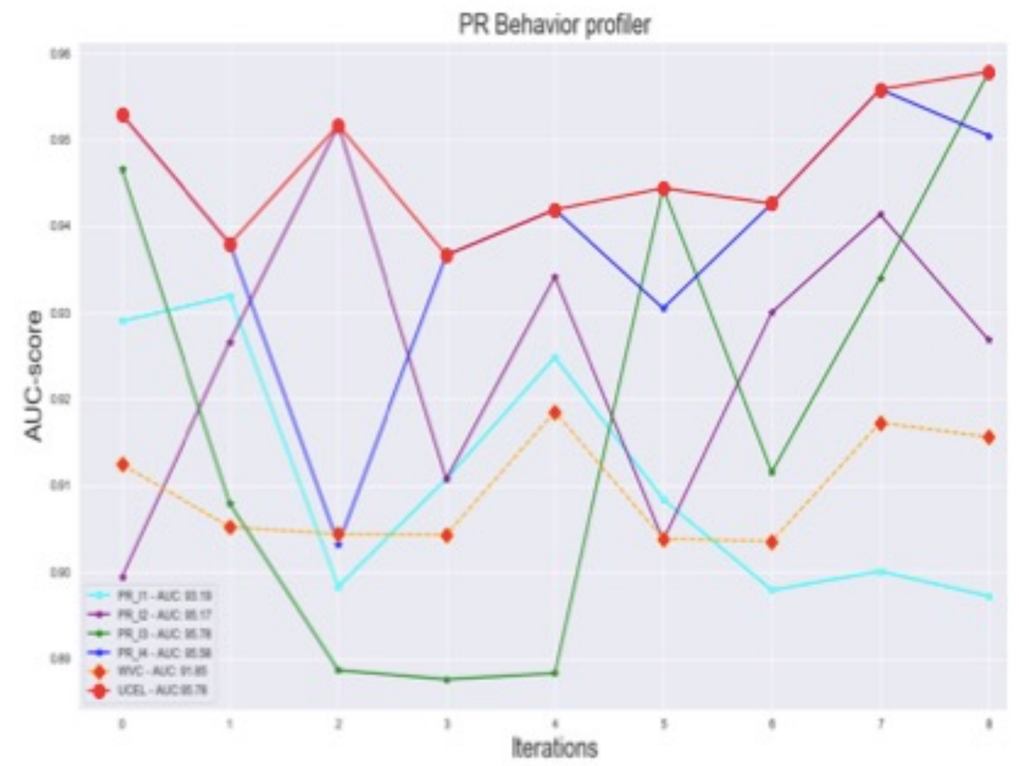


Evolution of the AUC score over the cycles with anomaly detection co-methods

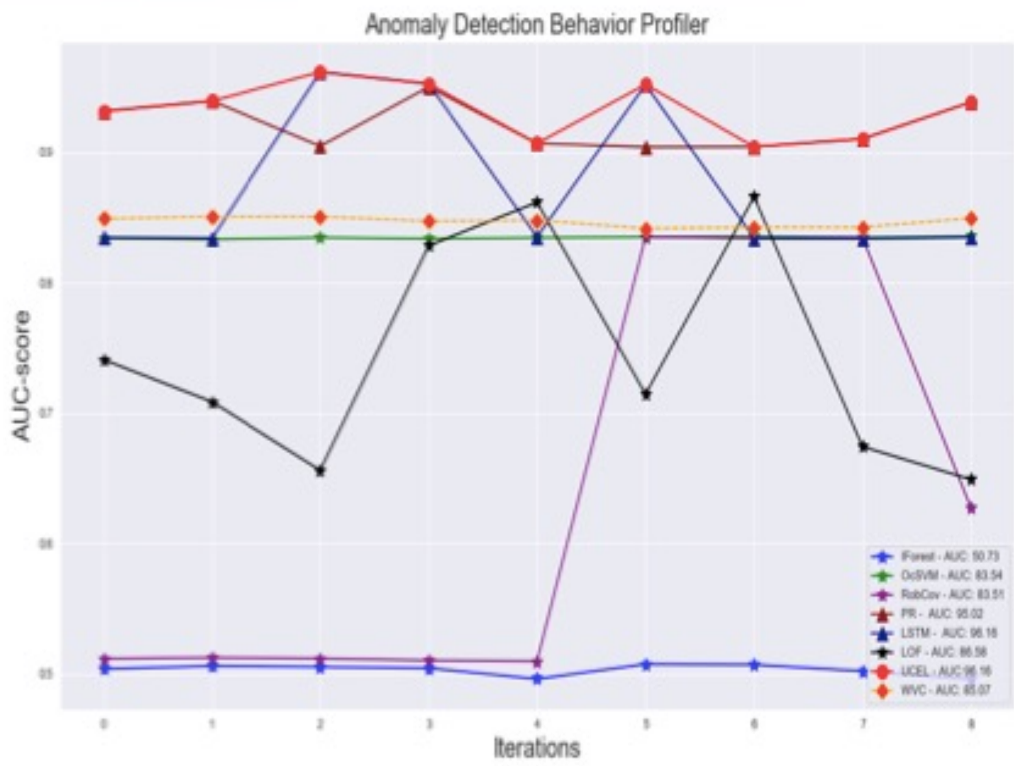
Left: MRobcov(10) vs individual boosted Robcov,
Right: 4 different anomaly detection methods vs individual boosted co-methods

UCEL performance

Evolution of the AUC score over the cycles with focus on graph-based anomaly detection using PageRank co-methods.



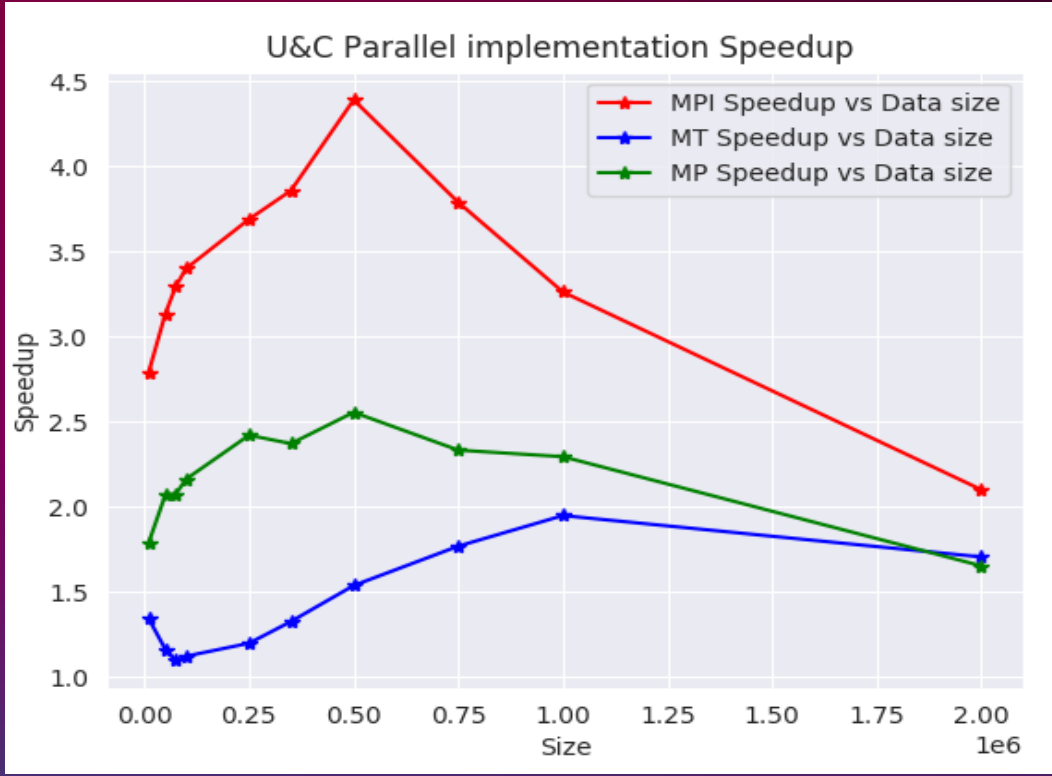
Multiple PR(4)

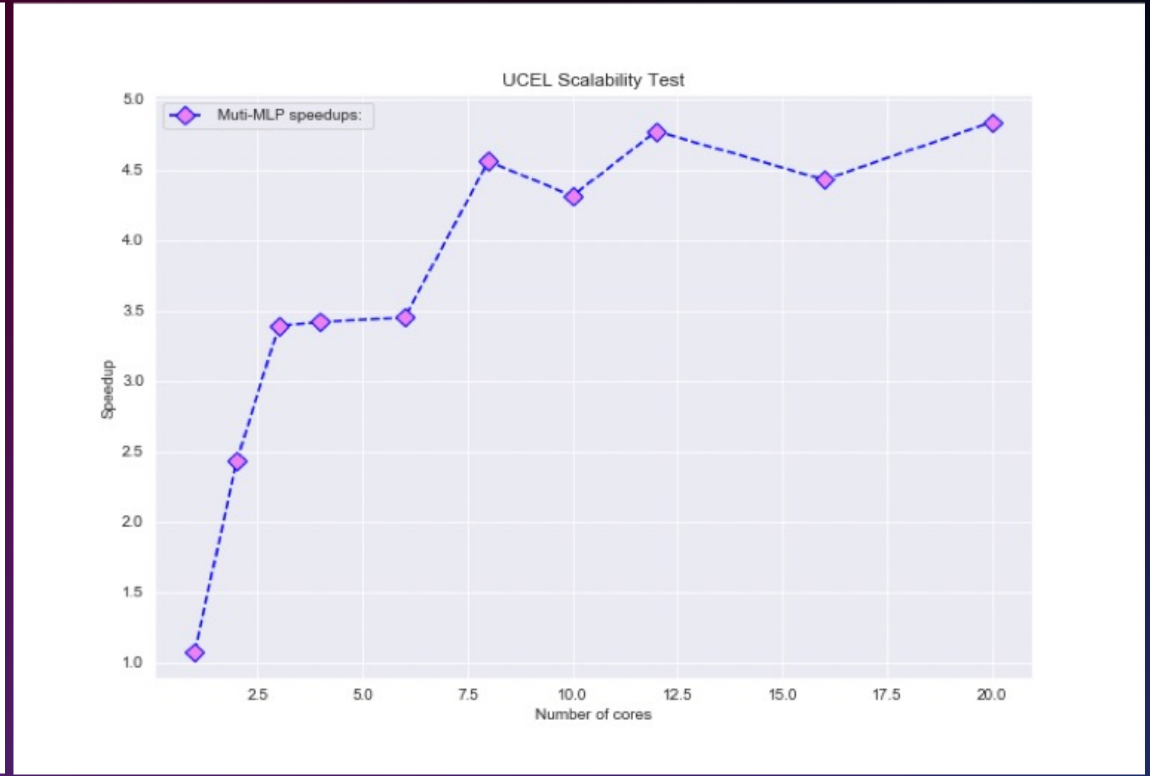


BP (4AD, PR, LSTM)

UCEL performance

Asynchronous communication: week & strong scalability



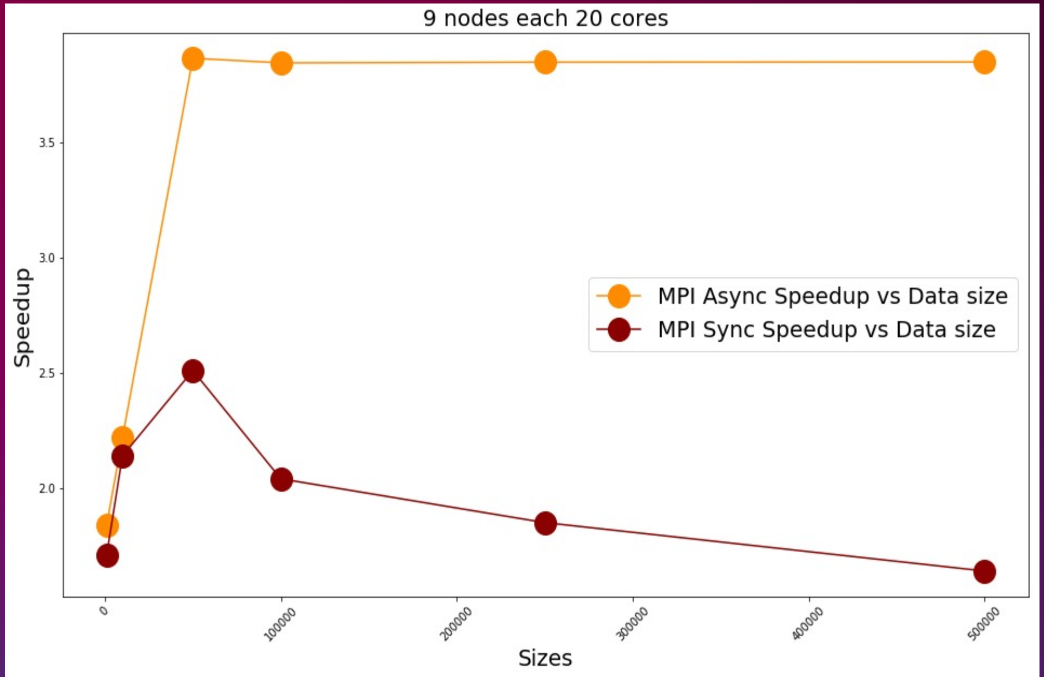


Weak scalability: due to the synchronization steps in the end of each cycle.

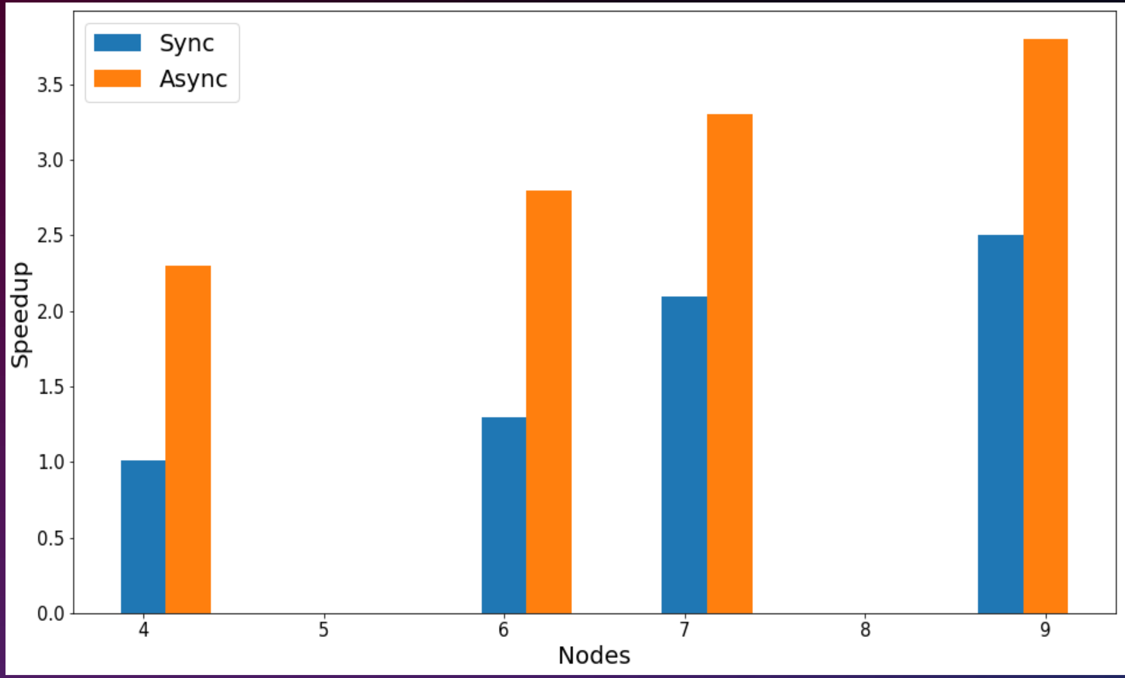
Strong scalability: the strong scalability of 10 MLP as co-methods, a dataset size of 500000 entries. The speedup increases from 1 to approximately 4.5 as the number of cores increases.

UCEL performance

Impact of asynchronous and synchronous communication between co-methods on performance



Asynchronous vs Synchronous model speedup



Max speedup as a function #nodes

Outline

- Main problems in linear algebra (moderate size)
- Large and sparse linear algebra problem
- High-performance AI and LA with applications
- Concluding remarks

Concluding Remarks

- Important impact of hybrid HP AI & LA : Be aware of not always using the libraries.
- Interactions between machine learning and linear algebra approaches must be studied more.
    - UCEL is a good example, and the approach is extensible.
    - Well adapted to high-performance parallel/distributed supercomputers
- ML presents a formidable tool bringing a tsunami of solutions to many problems
    - Experiment data (even when it does not seem interesting) must be saved...